Sécurité des serveurs et des applications, Partie 2

Matthieu Dien (matthieu.dien@unicaen.fr)

Ce qu’on va voir

- Principes de sécurité
- Exploits binaires
- Exploits web client
- Exploits web serveurs

Principes de sécurité

Tout ce que l’utilisateur peut modifier est dangereux

Dans un shell

Dans un shell

- les entrées utilisateurs (arguments de commande)
- les variables d’environnement ($PATH, etc)
- les fichiers/flux manipulables par l’utilisateur (pipe “|”, cat, mktemp, etc)
- les liens créés par l’utilisateur (ln -s)

Dans un binaire

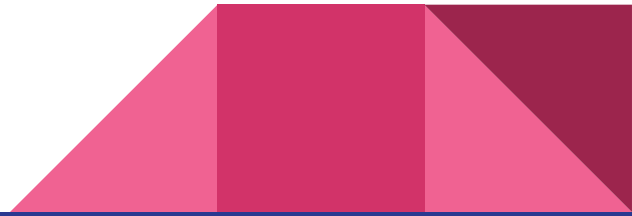
Dans un binaire

- les librairies dynamiques ($LD_PRELOAD)
- le flux d’exécution (gdb, ptrace)
- les entrées utilisateurs (argv, stdin)

Sur le web

Sur le web

- entrées utilisateurs (formulaires, $_GET, $_POST, Http Referer)
- cookies ($_COOKIES)
- user agent
- contrôle total du code côté client (DOM, scripts javascript)



L’information n’est pas toujours là où on le croit

Fuites d’informations (canaux cachés)

Fuites d’informations (canaux cachés)

- timing attack
- fichiers log
- répertoires non protégés (sans .htacess)
- erreurs (exceptions, réponses HTTP)
- robots.txt

Shell

Bit SUID

Permet d’exécuter un fichier avec les droits du propriétaire de fichier.

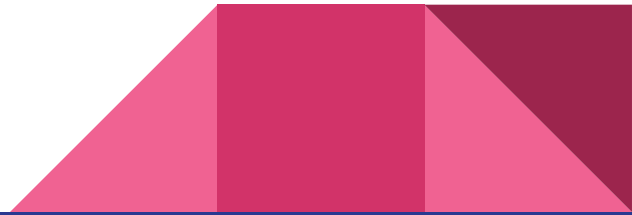
```
$ ls -l /bin/passwd
-rwsr-xr-x 1 root root 63624 31 juil. 21:12 /bin/passwd
```

Permet à tout utilisateur de pouvoir modifier le fichier /etc/shadow
=> une faille dans /bin/passwd et c’est la fin des haricots

Liens symboliques

```
$ ln -s /etc/shadow toto
$ ls -l

lrwxrwxrwx 1 matthieu matthieu 11 13 oct.  21:58 toto -> /etc/shadow
```

Permet de “berner” un programme sur le fichier en entrée.



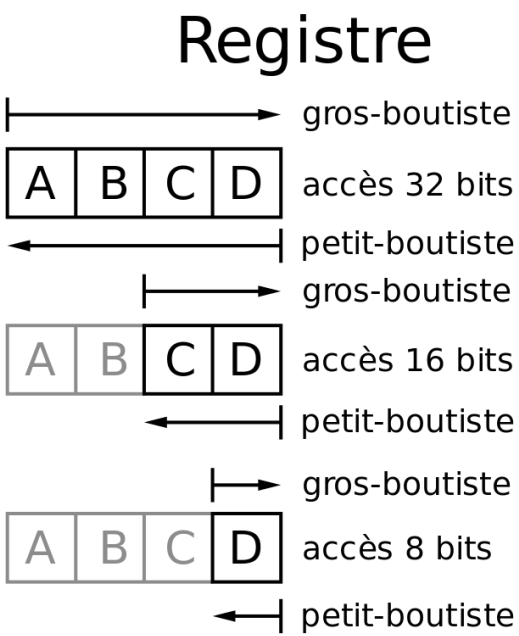
Variables d’environnements et sous-shell

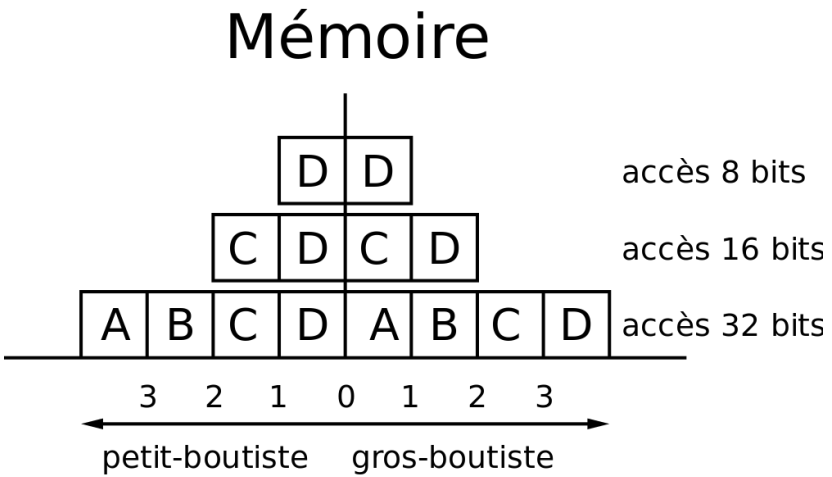
```
$ TOTO=toto
$ bash
$ echo $TOTO

$ exit
exit
$ export TOTO=toto
$ bash
$ echo $TOTO
toto
```

```
$ (TATA=tata ; yes) | echo $TATA

$ TATA=tata ; yes | echo $TATA
tata

$ TOTO=toto &
[1] 21546
$ echo $TOTO

[1]+  Fini                    TOTO=toto
$ TOTO=toto
$ echo $TOTO
toto
```

Exploits binaires

Big-endian / Little-endian





depuis Wikipedia par Konrad Eisele

Outils standards

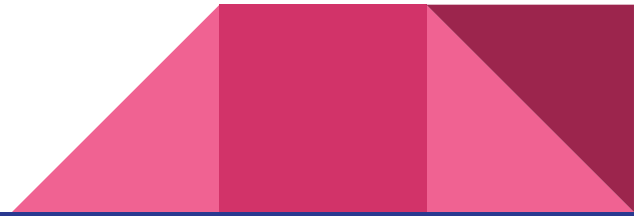
- strings (dump les chaînes ASCII d’un fichier)
- ltrace (trace les appels systèmes)
- gdb (GNU Debugger) :

  https://www.rocq.inria.fr/secret/Anne.Canteaut/COURS_C/gdb.html
- ptrace (appel système “process trace”)

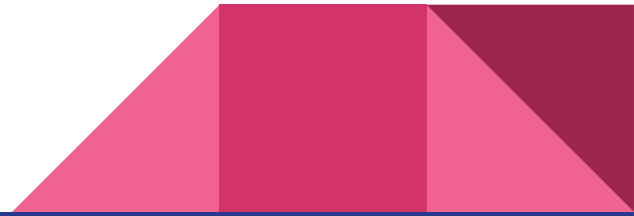
$LD_PRELOAD

```
$ ls
ma_lib.c ma_lib.h

$ gcc -c -fPIC ma_lib.c

$ gcc -shared ma_lib.o -o ma_lib.so

$ LD_PRELOAD=ma_lib.so un_executable
```

Buffer overflow

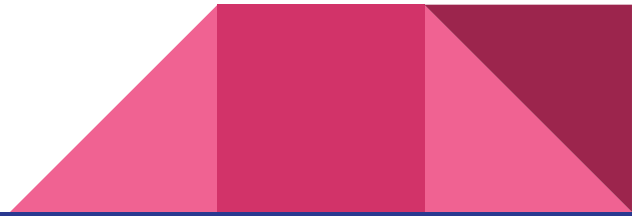


shellcode



format strings

- %x : affiche un unsigned int
- %s : affiche une chaîne de caractères (s’arrête de lire quand 0x00)
- %n : sauvegarde le nombre de caractères écrits jusqu’à présent



Programme de cette après-midi

- bandit (shell)
- leviathan (binaire débutant)
- narnia (binaire basique)
- behemot (je suis pas sûr de savoir tous les faire)